Fundamentos de Inteligência Artificial e Processamento de Linguagem Natural

O que é Inteligência Artificial (IA)?

Inteligência Artificial é um campo da ciência da computação focado em criar sistemas que simulam a capacidade humana de aprender, raciocinar e tomar decisões. A IA busca desenvolver máquinas que conseguem analisar dados, identificar padrões e executar tarefas de forma autônoma, sem a necessidade de serem programadas para cada passo específico.

Tipos de Aprendizado de Máquina

O cerne da IA moderna está no **Aprendizado de Máquina**, onde os modelos "aprendem" com os dados. Os principais tipos são:

- **Aprendizagem Supervisionada:** O modelo é treinado com um conjunto de dados que já possui rótulos (ou seja, as respostas corretas). É como aprender com um professor que sempre corrige os exercícios. Por exemplo, em nosso minicurso, vamos usar dados de tweets já classificados como positivos, neutros ou negativos para treinar nosso modelo.
- **Aprendizagem Não Supervisionada:** O modelo recebe dados sem rótulos e precisa encontrar padrões e estruturas por conta própria. É como agrupar objetos por cor ou tamanho sem que alguém diga qual é qual.
- **Aprendizagem por Reforço:** O modelo aprende através de tentativa e erro, recebendo "recompensas" ou "penalidades" por suas ações. É o tipo de aprendizado usado para treinar robôs a jogar xadrez ou videogames.

O que é Processamento de Linguagem Natural (PLN)?

O **Processamento de Linguagem Natural (PLN)**, ou **NLP** em inglês, é a área da IA que permite que computadores entendam, interpretem e gerem a linguagem humana. Vamos focar em fazer com que uma máquina "leia" e "compreenda" o sentimento de um tweet. Lembrando que Linguagem Natural é toda linguagem humana da comunicação desde a formalidade à informalidade, do texto ao áudio.

Dentro do PLN, existem duas subáreas importantes:

- **Geração de Linguagem Natural (GLN/NLG):** Focada em gerar texto a partir de dados estruturados. Por exemplo, um sistema que gera automaticamente resumos de relatórios financeiros.
- **Compreensão de Linguagem Natural (NLU/NLU):** Focada em extrair significado de um texto, como identificar a intenção em uma pergunta ou, no nosso caso, classificar o sentimento de um tweet.

Ramos e Tecnologias de PLN

O PLN é um campo vasto. Tradicionalmente, usávamos técnicas baseadas em regras e estatísticas. Atualmente, os modelos mais avançados usam **redes neurais** e **transformadores** (como o BERT e o GPT-3), que são capazes de capturar contextos complexos.

Aplicações

Por que não usamos sempre as tecnologias mais avançadas?

Esta é uma excelente pergunta! O avanço tecnológico é constante, mas nem sempre a solução mais recente é a ideal. Optar por técnicas mais tradicionais de PLN pode ser a melhor escolha por várias razões:

- **Custo e Acessibilidade:** Modelos de ponta como o GPT-3 exigem um poder computacional gigantesco e podem ser caros de usar. As técnicas tradicionais, como as que usaremos, podem rodar em qualquer máquina, incluindo um notebook simples.
- **Complexidade e Tempo:** A implementação de modelos de ponta pode ser mais complexa e demorada. Técnicas tradicionais são mais fáceis de entender, implementar e depurar, o que é ideal para começar.
- **Interpretabilidade:** É mais fácil entender por que um modelo tradicional tomou uma decisão. Os modelos mais avançados são frequentemente "caixas-pretas," onde é difícil explicar a lógica por trás de suas previsões.

O ideal, como em tudo na vida, é escolher as ferramentas certas para a situação certa. Como podemos concluir, usar o ferramental mais desenvolvido e complexo pode ser um “exagero” em alguns aspectos. Isso vai depender do contexto e é responsabilidade de um profissional da área avaliá-lo.

---

2. Nosso Projeto Prático: Classificação de Tweets

O que iremos fazer?

Neste minicurso, vamos construir um modelo de Aprendizado de Máquina que aprende a partir de dados de tweets sobre companhias aéreas. O objetivo final é que nosso modelo seja capaz de **classificar corretamente um novo tweet** como **positivo**, **neutro** ou **negativo** com uma certa confiança, que analisaremos ao final.

Ferramenta: Google Colab

Usaremos o **Google Colab**, uma plataforma gratuita baseada na nuvem que permite escrever e executar código Python diretamente no seu navegador. Ele já vem com as principais bibliotecas de IA e ciência de dados instaladas.

Ele é composto por células que podem ser de código ou textuais. Em geral é bem intuitivo para que já domina a linguagem Python e bem flexível.

Cada Célula de código pode ser executada por si só. No entanto, devemos atentar-nos aos recursos. Se deseja utilizar um recurso da célula 1 na célula 2, mas a célula 1 ainda não foi executada, a célula 2 não terá acesso ao recurso ou terá o recurso desatualizado.

Revisando os Imports

Para nosso projeto, vamos usar algumas bibliotecas essenciais:

- **Pandas:** Para carregar, manipular e analisar nossos dados em formato de tabela (DataFrame).
- **Matplotlib e Seaborn:** Para criar visualizações e gráficos que nos ajudarão a entender melhor nossos dados e os resultados do modelo.
- **Scikit-learn:** A principal biblioteca de aprendizado de máquina em Python. Usaremos ela para pré-processar os dados, treinar e testar nosso modelo.
- **NLTK (Natural Language Toolkit):** Uma biblioteca clássica para PLN, que nos ajudará com tarefas de pré-processamento de texto, como a remoção de *stopwords*.
- **WordCloud:** Para gerar visualizações em nuvem de palavras, o que nos ajuda a identificar as palavras mais frequentes nos tweets.
- **Regex (Expressões Regulares):** Para limpar o texto de caracteres indesejados, como menções e links.

Obtenção e Estrutura dos Dados

Vamos trabalhar com um **DataFrame** de tweets sobre companhias aéreas. Cada linha do DataFrame contém um tweet e sua respectiva classificação: `positivo`, `neutro` ou `negativo`. Nosso desafio é usar esses dados para treinar um modelo que possa prever a classificação de novos tweets.

---

3. Limpeza e Pré-processamento de Texto

Antes de qualquer modelo de aprendizado de máquina conseguir entender nosso texto, precisamos limpá-lo e prepará-lo. Esta etapa é crucial para o sucesso do nosso projeto.

-**Undersampling**:

Técnica que reduz o número de amostras da classe majoritária para equilibrar a distribuição das classes. É útil quando se deseja evitar que o modelo seja tendencioso em favor da classe com maior representatividade.

-**Oversampling**:

Técnica que aumenta o número de amostras da classe minoritária, seja por duplicação ou geração de amostras sintéticas. É eficaz para garantir que o modelo tenha uma representação adequada de ambas as classes.

-**Underfitting**:

Ocorre quando o modelo é muito simples e não consegue capturar a complexidade dos dados. Ele apresenta baixa performance tanto nos dados de treino quanto nos de teste, pois não aprende adequadamente.

-**Overfitting**:

Acontece quando o modelo é excessivamente complexo e aprende muito bem os dados de treino, mas não generaliza bem para dados novos, resultando em boa performance no treino e baixa nos testes.

Limpeza e Pré-processamento

Neste material, por questões didáticas, irei exemplificar no contexto do portugês, mas o DataFrame utilizado está em inglês.

Primeiramente, vamos refinar nosso texto com as seguintes etapas:

- **Converter para Caixa Baixa (**`to lower case`**):** Deixar todas as letras minúsculas (`'Ola'` e `'ola'` viram `'ola'`). Por que fazemos isso? Para que o modelo trate as mesmas palavras, independentemente de estarem no começo da frase ou não, como a mesma entidade. Isso evita que `'Bom'` e `'bom'` sejam tratados como palavras diferentes.
- **Remover Stopwords: Stopwords** são palavras comuns na língua (como "o", "a", "de", "e") que geralmente não carregam muito significado por conta própria. Remover essas palavras ajuda o modelo a focar nas palavras que realmente importam para o sentimento.
- **Limpar Dados Específicos do Twitter:**
    - **Links:** Remover URLs, pois eles não contribuem para o sentimento do tweet.
    - **Menções:** Remover menções de usuários (`@usuario`), pois elas raramente afetam o sentimento geral do texto.
- **Remover Caracteres Especiais e Números:** Focar apenas nas palavras, texto puro.
- **Remover Espaços Excessivos:** Tratar múltiplos espaços em branco, garantindo que tudo fique padronizado.

Tokenização

Consiste no processo de dividir um texto em unidades menores, chamadas **tokens**.

- **O que são tokens?** Eles podem ser palavras, sentenças ou até mesmo subpalavras. A forma mais comum de tokenização é transformar um texto contínuo em uma lista de palavras. Por exemplo, a frase "pessimo atendimento banheiro sujo demais" pode virar a lista de tokens pós processamento inicial: `['pessimo', 'atendimento', 'banheiro', 'sujo', 'demais']`.
- **Quando usar?** Sempre! A tokenização é a base para todas as etapas seguintes.

Tipos de Tokenização

Existem várias formas de tokenizar um texto, mas as mais comuns são:

- **Por Espaço em Branco:** A mais simples, mas pode falhar com pontuação.
- **Por Pontuação:** Considera a pontuação como um token separado, o que é mais robusto.
- **Específica para Mídias Sociais:** Ideal para nossos tweets, pois lida com elementos como hashtags (`#pln`), menções (`@usuario`) e emojis, mas não o caso se limparmos toda informação no pré processamento.

Stemming e Lematização

Essas duas técnicas servem para reduzir as palavras à sua forma base, o que ajuda a padronizar o vocabulário (uma ou outra é aplicada).

- **Stemming:** Reduz a palavra ao seu "radical," cortando sufixos e prefixos. Por exemplo, "correndo", "corria" e "corre" podem se tornar "corr". É mais rápido, mas pode gerar palavras que não existem.
- **Lematização:** Reduz a palavra ao seu "lema" (sua forma canônica) usando um dicionário. Por exemplo, "correndo", "corria" e "corre" se tornam "correr". É mais lento, mas mais preciso.

Para o nosso minicurso, a lematização é geralmente a melhor escolha, mas ambas são opções válidas.

---

4. Representação Numérica (Vetorização)

Nossos modelos de aprendizado de máquina só entendem números, não texto. A **vetorização** é o processo de converter nossos tokens de texto em vetores numéricos.

Bag of Words (Saco de Palavras)

O **Bag of Words** é uma técnica simples e eficaz.

- **Como funciona:** Ele cria um "saco de palavras" com todas as palavras únicas de todos os tweets. Cada tweet é então representado por um vetor, onde cada posição corresponde a uma palavra do saco. O valor da posição pode ser a contagem de vezes que a palavra aparece naquele tweet. Diferente de conjuntos, pois permite a repetição de elementos.
- **Vantagem:** É simples e rápido de implementar. Ideal para tarefas de classificação onde a frequência das palavras é um bom indicador.
- **Limitação:** Ele ignora completamente a ordem das palavras, o que pode fazer com que o modelo perca o contexto.

TF-IDF (Term Frequency-Inverse Document Frequency)

O **TF-IDF** é uma versão mais sofisticada do Bag of Words.

- **Como funciona:** Ele não apenas conta a frequência de uma palavra (**Term Frequency**), mas também dá um peso maior a palavras que são raras e importantes em nosso conjunto de dados (**Inverse Document Frequency**). Por exemplo, a palavra "ótimo" terá um peso maior do que a palavra "avião", pois "ótimo" é muito mais distintiva para um tweet positivo.
- **Vantagem:** É uma técnica muito popular e robusta que melhora a classificação ao focar nas palavras mais relevantes.
- **Limitação:** TF-IDF depende do tamanho do corpus. Se ele for pequeno, o modelo fica menos confiável.

- **Ele desconsidera o contexto da palavra — analisa as palavras de forma isolada.**
-

Word Embeddings (Representações de Palavras)

Os **Word Embeddings** são a base dos modelos de ponta.

- **Como funciona:** Eles representam cada palavra como um vetor denso (uma lista de números decimais), onde a posição no espaço vetorial captura relações semânticas e sintáticas. Por exemplo, a distância entre o vetor de "rei" e o vetor de "homem" é similar à distância entre o vetor de "rainha" e o de "mulher".
- **Vantagem:** Capturam o significado e o contexto das palavras de forma muito mais rica.
- **Limitação:** São mais complexos de gerar e entender, e geralmente exigem mais dados.

Para o nosso minicurso, vamos focar em Bag of Words ou TF-IDF, pois são ideais para começar e nos darão excelentes resultados.

---

5. Algoritmos Classificadores e Análise de Resultados

Algoritmos Classificadores

Depois de vetorizar nossos dados, precisamos de um algoritmo que aprenda com eles. Existem muitos, mas alguns dos mais tradicionais e eficazes para classificação de texto incluem:

- **Naive Bayes:** é um algoritmo probabilístico que se baseia no Teorema de Bayes e assume que as características (features) de um dado são independentes umas das outras. Para classificar um novo dado, ele calcula a probabilidade de o dado pertencer a cada classe, assumindo que a presença de uma característica não afeta a de outra. A classe com a maior probabilidade é então atribuída ao dado.
- **KNN(K-Nearest Neighbors)**:é um algoritmo de aprendizado de máquina não paramétrico. Ele não aprende um modelo a partir dos dados de treinamento; em vez disso, ele armazena os dados e usa a distância entre o novo dado e seus vizinhos mais próximos (os "K" vizinhos) para fazer a classificação. Se a maioria dos K vizinhos mais próximos for da classe A, o novo dado é classificado como A. A escolha do valor de K é crucial, pois um K muito pequeno pode tornar o modelo sensível a ruídos, enquanto um K muito grande pode suavizar as fronteiras de decisão e ignorar nuances importantes.

Dado os exemplos, nesse minicursos será explorado o Naive Bayes. Com isso, temos suas implicações:

O Naive Bayes é um classificador probabilístico popular, especialmente para tarefas de classificação de texto. No entanto, seu nome já entrega sua principal característica e limitação: a suposição de "ingenuidade" (*naive*).

A Premissa "Ingênua" e Suas Consequências

A fórmula do Naive Bayes se baseia na suposição de que **os atributos são condicionalmente independentes** dado a classe. Em outras palavras, ele assume que a presença de uma palavra em um documento (por exemplo, a palavra "bom") não afeta a probabilidade de outra palavra (por exemplo, a palavra "serviço") aparecer, dado que o documento pertence a uma classe específica (por exemplo, "positivo").

- **Implicação:** Essa suposição raramente é verdadeira na linguagem natural. Palavras aparecem em conjunto; "excelente serviço" é uma combinação muito mais provável do que "excelente banana" em um contexto de avaliação de serviços.
- **Consequência:** Apesar dessa suposição ser falsa, o Naive Bayes frequentemente tem um desempenho surpreendentemente bom. A razão é que, para tarefas de classificação, o que realmente importa não é a precisão das probabilidades absolutas, mas sim a ordem relativa das probabilidades. O Naive Bayes muitas vezes consegue classificar corretamente a classe mais provável, mesmo que as probabilidades calculadas não sejam perfeitamente precisas.

A Questão da Probabilidade Zero e a Suavização

Sua explicação sobre a suavização de Laplace é precisa e crucial. Sem ela, o algoritmo se torna extremamente frágil.

- **Implicação:** se uma palavra aparece em um novo texto, mas nunca foi vista na classe de treinamento, a probabilidade para essa classe se torna zero. Isso anularia completamente a capacidade do modelo de fazer uma previsão razoável.
- **Consequência:** A **suavização** é uma técnica obrigatória ao usar Naive Bayes com texto. Ela evita que a probabilidade zero cause um colapso na predição. Ao adicionar um pequeno valor (α) ao numerador e um valor ajustado ao denominador, a suavização garante que mesmo palavras não vistas tenham uma probabilidade pequena, mas não nula. Isso permite que o modelo ainda considere o peso dos outros termos no cálculo final. A suavização, portanto, não apenas corrige um problema matemático, mas também melhora a robustez do modelo a novas palavras.

Outras Vantagens e Limitações

Vantagens:

- **Simplicidade e Eficiência:** O Naive Bayes é extremamente rápido para treinar e fazer previsões. Ele exige poucos dados de treinamento em comparação com modelos mais complexos e é fácil de implementar.
- **Bom Desempenho com Pequenos Dados:** Ele pode funcionar bem mesmo com um conjunto de dados de treinamento relativamente pequeno, o que o torna uma ótima escolha para projetos iniciais ou com recursos limitados.

Limitações:

- **Não Lida com Relações Complexas:** Por causa da suposição de independência, o Naive Bayes não consegue capturar a ordem das palavras ou as relações entre elas (por exemplo, a diferença entre "bom não" e "não bom").
- **Vieses de Dados:** O modelo é sensível à distribuição dos dados de treinamento. Se uma classe for muito mais frequente do que outra, o modelo pode ter um viés para a classe mais comum.

Treinamento e Teste do Modelo

Vamos dividir nossos dados em duas partes:

- **Conjunto de Treinamento:** Usaremos a maior parte dos dados para treinar o modelo e deixá-lo aprender os padrões.
- **Conjunto de Teste:** Usaremos uma parte dos dados que o modelo nunca viu para testá-lo e ver se ele generaliza bem. Isso nos dará uma métrica mais realista de como nosso modelo se sairia no mundo real.

Análise de Resultados

Ao final, usaremos métricas como a **matriz de confusão**,além de **acurácia, precisão, cobertura, medida-f** para entender o desempenho do nosso modelo.

A matriz de confusão nos mostra visualmente quantos tweets foram classificados corretamente e quantos foram incorretamente, permitindo-nos identificar os pontos fortes e fracos do nosso modelo. Já os demais são medidas clássicas para avaliar os classificadores.

Medidas de avaliação

- **Matriz de confusão:** nos mostra visualmente quantos tweets foram classificados corretamente e quantos foram incorretamente, permitindo-nos identificar os pontos fortes e fracos do nosso modelo. Já os demais são medidas clássicas para avaliar os classificadores.

- **Acurácia (accuracy):** indica a fração de acertos do classificador. Quanto maior a acurácia, maior a taxa geral de acertos.

$$\frac{VP + VN}{VP + FP + VN + FN}$$

- **Precisão (precision):** indica a fração das mensagens classificadas como spam que, de fato, eram spam. Um classificador com alta precisão raramente põe um dado em uma classe a que ele não pertence.

$$\frac{VP}{VP + FP}$$

- **Cobertura (recall):** indica a fração das mensagens que, de fato, eram spam e que foram classificadas como tal. Um classificador com boa cobertura raramente deixa de incluir um dado em uma classe a que ele pertence.

$$\frac{VP}{VP + FN}$$

- **Medida-F (F-Score):** combina precisão e cobertura.

$$F = 2 \times \frac{\text{precisão} \times \text{cobertura}}{\text{precisão} + \text{cobertura}}$$

Referências:

- FERREIRA, Marcelo; LOPES, Marcos. Linguística Computacional. São Paulo: Contexto, 2021.

- Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português: https://brasileiraspln.com/livro-pln/

- WordCloud: https://amueller.github.io/word_cloud/

- Scikit-learn: https://scikit-learn.org/stable/

- Regex: https://www.w3schools.com/python/python_regex.asp

- Seaborn: https://seaborn.pydata.org/

- Matplotlib: https://matplotlib.org/stable/index.html

- Pandas: https://pandas.pydata.org/docs/